

REMARKS/ARGUMENTS

Claims 1-27 are pending in the application and have been rejected. Claims 1, 2, 6, 8, 11, 18, 19, 22, 23, 26, and 27 have been amended.

Rejections under 35 U.S.C. §112.

Claims 2-13, 19-21, and 23-27 were rejected under 35 U.S.C. §112, second paragraph, as being indefinite. Specifically, the Office Action contends that in claims 2, 19, 21, 23, 26, and 27 the term "the status" lacks antecedent. The claims have been amended to correct the antecedent.

Rejections under 35 U.S.C. §103

The Office Action rejected claims 1-9, 14, and 18-27 under 35 U.S.C. §103(a) as being unpatentable over Adl-Tabatabai et al [US Patent No. 6,317,869], in view of Jagannathan et al [US Patent No. 5,692,193]. Applicant hereby requests reconsideration of the rejection in view of the amendments and remarks made herein.

Claim 1 has been amended to read as follows:

A method of managing memory in a multi-threaded processing environment including respective local thread stacks and heaps and a global heap, said method comprising:

- creating an object in a thread heap;
- monitoring the object to determine whether the object is referenced only from a given thread stack;
- assigning a status to the object; and
- changing the status of the object under certain conditions.

The Office Action contends that Adl-Tabatabai discloses the invention as claimed except that it does not disclose monitoring the object as claimed. However, the Office Action concedes that Adl-Tabatabai does not disclose monitoring the object to determine whether the object is referenced only from a given thread stack. However, the Office Action contends that Jagannathan teaches this element and that it would have been obvious to one skilled in the art to

modify Adl-Tabatabai according to Jagannathan. Applicant respectfully submits that, as amended, the claims at issue would not have been obvious in view of the combination of Adl-Tabatabai and Jagannathan.

To establish obviousness the combination of references must not only disclose each element of a claim but must also provide a motivation or reason to combine the references. In this case, the Office Action contends that the motivation to combine is as follows:

Adl-Tabatabai has no teachings relevant to the claimed subject matter. Adl-Tabatabai relates to a method of run-time tracking of object references that uses a bit vector for an ambiguously typed variable and maintaining a bit for determining whether the variable is assigned a value. See claim 1. The Adl-Tabatabai patent appears to be a solution of the problem of mistakenly considering a value as a reference value in a garbage collection process. That has nothing to do with the claimed invention which relates to assigning of status for objects as either local or global.

Jagannathan makes a decision on locality once and does not change it once the decision is made. Jagannathan neither teaches nor suggests changing the status of an object as claimed and does not provide any motivation for combining the references. Certainly, the Office Action rationale for making the combination is neither taught nor suggested by Jagannathan or the combination of the cited references. There is no mechanism in Jagannathan for tracking the locality of objects; Jagannathan specifically states the rules which govern the placement of objects in the stack, thread local heap and global heap. The placement decision is made once and not subsequently altered. The claimed mechanism by contrast specifically teaches the mechanism by which an object can be assigned to a local heap and then moved to the global heap under certain circumstances such as when it becomes globally reachable. Therefore, one skilled in the art would not have been motivated to modify the references as suggested in the Office Action.

Applicants provide the following explanation to aid in understanding the claimed invention. However, it is not intended to limit the scope of the claims. Thread private or local heaps are used to allocate non-shared objects whose lifetimes might exceed the lifetime of the procedures that created them. The term "might exceed" is used because it is not always possible for the compiler to determine the lifetime of an object in programming languages such as Scheme or ML. Furthermore, it may not be possible to determine the lifetimes of objects in languages which allow calls to unknown procedures. References contained in private heap can refer to other objects in the same private heap, or objects in shared or global heaps, but they cannot refer to objects in the stack. References in the stack may refer to objects in the private heap, but references in the shared heap may not. Private heaps lead to greater locality since data allocated on them are used exclusively by a single thread of control; the absence of interleaving allocations among multiple threads means that objects close together in the heap are likely to be logically related to one another.

No other thread can access objects that are contained in a thread's stack or local heap. Thus, both thread stacks and local heaps can be implemented in local memory on the processor without any concern for synchronization or memory coherency. Thread local heaps are actually a series of heaps organized in a generational manner. Storage allocation is always done in the youngest generation in a manner similar to other generational collectors. As objects age they are moved to older generations. All garbage collection of the local heap is done by the thread itself. In most thread systems that support garbage collection all threads in the system must be suspended during a garbage collection.

In contrast to the claimed approach, Jagannathan's threads garbage collect their local heaps independently and asynchronously with respect to other threads. Thus other threads can continue their computation while any particular thread collects its local heap; this leads to better load balancing and higher throughput. A second advantage of this garbage collection strategy is that the cost of garbage collecting a local heap is charged only to the thread that allocates the storage, rather than to all threads in the system.

The Office Action contends that the limitation of claim 3 (deleting from the thread heap one or more local objects when it is determined that they are not accessible from a local root) is disclosed at Adl-Tabatabai col. 7, lines 29-46. That section relates to garbage collection and does not suggest the claimed combination. Moreover, claim 3 is distinguishable over the cited references for the same reasons that claim 1 is distinguishable.

The Office Action contends that the limitation of claim 5 (changing the status of an object in the thread heap to global if the object is assigned to a static variable or if the object is assigned to a field in a global object) is disclosed at Adl-Tabatabai col. 5, lines 29-30 and 43-44. As discussed above, Adl-Tabatabai neither teaches nor suggests changing the local/global status of an object. Moreover, claim 5 is distinguishable over the cited references for the same reasons that claim 1 is distinguishable.

The Office Action contends that the limitation of claim 6 (intercepting assignment operations to an object in the thread heap) is disclosed at Adl-Tabatabai item 540 in fig. 5B. Item 540 is the mere assignment of a value to a variable at the beginning of a method; it does not relate to changing the status. As discussed above, Adl-Tabatabai neither teaches nor suggests changing the local/global status of an object. Moreover, claim 6 is distinguishable over the cited references for the same reasons that claim 1 is distinguishable.

The Office Action contends that the limitation of claim 16 (analysing whether an object is likely to be made global and associating such an object with a global status on creation) is disclosed at Adl-Tabatabai col. 4, line 58 – col. 5, line 9). That section says nothing about the *likelihood* of an object being global. Moreover, claim 16 is distinguishable over the cited references for the same reasons that claim 1 is distinguishable.

The Office Action rejected claims 10-13 and 15-17 as unpatentable over Adl-Tabatabai in view of Jagannathan and further in view of United States Patent 6,308,315 to Dice *et al.*

(hereafter, Dice).

The Office Action admits that the limitation of claim 10 (using spare capacity in an object header for the status) is not disclosed by either Adl-Tabatabai or Jagannathan but that the subject matter of claim 10 would have been obvious in view of Dice col. 6, lines 35-38. That section of Dice discusses object headers having space for variables – a well known general fact – but says nothing about using spare capacity for status. In addition, the rationale used in the Office Action for combining three disparate references was that the combination would make garbage collection faster and more precise and this is not supported by any evidence of record. Applicant therefore requests evidence for that assertion. Moreover, claim 10 is distinguishable over the cited references for the same reasons that claim 1 is distinguishable.

The Office Action contends that the limitation of claim 11 (using multiples of 2 or more bytes in a thread heap to store the objects) is disclosed at Adl-Tabatabai col. 5, lines 5-7. However, the Office Action does not even allege that the limitation “whereby there is at least one spare bit in the object length variable and using the at least one spare bit as the status” is disclosed anywhere in the cited references. Moreover, claim 11 is distinguishable over the cited references for the same reasons that claim 1 is distinguishable.

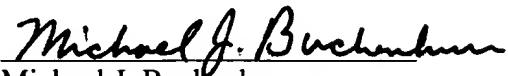
The Office Action contends that the limitation of claim 12 (moving objects whose status is global from the thread heap to the global heap) is disclosed at Jagannathan col. 14, lines 39-42. Applicant respectfully traverses this statement. Jagannathan neither teaches nor suggests this limitation. The cited portion discusses moving half of the TCBs in the local pool to the global pool when the local pool overflows. That has nothing to do with the claimed limitation.

The Office Action contends that the limitation of claim 13 (compacting the reachable local objects in a thread heap) is disclosed at Adl-Tabatabai col. 2, lines 30-32. Applicant respectfully traverses this statement. That section discusses compacting memory space by moving objects it finds to another region. It says nothing about the thread heap.

Claims 4, 7-9, 14, 15-17, and 18-27 are distinguishable over the cited combination of references for reasons discussed above.

For the foregoing reasons, Applicant respectfully requests allowance of the pending claims and that a timely Notice of Allowance be issued in this case.

Respectfully submitted,


Michael J. Buchenhorner
Reg. No. 33,162

Date: November 16, 2003

HOLLAND & KNIGHT LLP
Holland & Knight LLP
701 Brickell Avenue, Suite 3000
Miami, FL 33131
(305) 789-7773 (voice)
(305) 789-7799 (fax)